

Visibility Silhouettes for Semi-Analytic Spherical Integration

Derek Nowrouzezahrai^{1,2} Ilya Baran^{2,3} Kenny Mitchell² Wojciech Jarosz²

¹Université de Montréal ²Disney Research Zurich ³Belmont Technology, Inc.



Figure 1: We compute spherical integrals of visibility-masked functions commonly found in rendering, e.g. in ambient occlusion and all-frequency direct illumination. This scene (123.2k triangles) took ~ 5 minutes to render on a 4-core CPU, independent of the lighting. These results render at 84% to 106% the speed of a noise-free ray-traced image, depending on the lighting.

Abstract

At each shade point, the spherical visibility function encodes occlusion from surrounding geometry, in all directions. Computing this function is difficult and point-sampling approaches, such as ray-tracing or hardware shadow mapping, are traditionally used to efficiently approximate it. We propose a semi-analytic solution to the problem where the spherical silhouette of the visibility is computed using a search over a 4D dual mesh of the scene. Once computed, we are able to semi-analytically integrate visibility-masked spherical functions along the visibility silhouette, instead of over the entire hemisphere. In this way, we avoid the artifacts that arise from using point-sampling strategies to integrate visibility, a function with unbounded frequency content. We demonstrate our approach on several applications, including direct illumination from realistic lighting and computation of PRT data. Additionally, we present a new frequency-space method for exactly computing all-frequency shadows on diffuse surfaces. Our results match ground truth computed using importance-sampled stratified Monte Carlo ray-tracing, with comparable performance on scenes with low-to-moderate geometric complexity.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and realism—Color, shading, shadowing, and texture

1. Introduction

Shadows are one of the most identifiable real-world lighting phenomena, providing important depth cues and information about the surrounding lighting environment. Advances in rendering are increasing the accuracy and performance of shadow generation, allowing *all-frequency* shadows from real-world lighting on scenes of growing complexity.

While initial work on high-performance realistic shadows focused on simple point and directional light sources (see recent surveys [ESAW11, WP12]), the availability of real-world data has prompted generalizations of these approaches to more complex lighting. Efficiently computing accurate shadows from environment lights is a difficult problem because, at each shading point, light must be integrated from

all directions and masked by the *spherical binary visibility* function (or *visibility*, for brevity) at each shade point.

Computing visibility is often the bottleneck in realistic rendering algorithms. Only unoccluded light directions contribute to direct illumination, after weighting by the view-evaluated BRDF. Computing the visibility is a difficult geometric problem (cf. [DDP02]), typically solved with sampling approaches, such as ray-tracing. Recent GPU ray-tracing systems can accelerate this costly computation.

We propose a geometric solution to this problem and exploit the fact that, for many scenes, visibility can be compactly represented by its (spherical) boundary, or *silhouette*. We do not consider scenes where this assumption breaks down (e.g., stacked/jittered chain-linked fences, complex foliage,

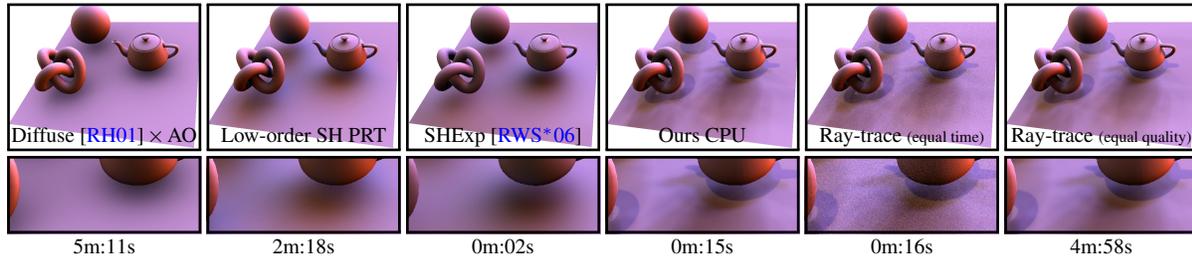


Figure 2: We compare common approaches for handling visibility. For a noise-free ray-traced result, our approach outperforms importance-sampled QMC on this simple test scene (37.8k triangles). PRT is computed per-vertex using QMC. QMC scales better with scene complexity, whereas our approach scales favorably with spherical sampling rate (see discussion in Section 5).

etc.); our method would still work on such scenes, but would suffer from reduced performance. We compute the spherical geometric contours about a shade point using a 4D acceleration structure, exploiting the fact that only a small percentage of all triangle edges contribute to these spherical contours. We then use these contours to accurately compute semi-analytic integrals of visibility-masked functions.

Integrals of visibility-masked functions appear in many rendering applications, such as direct lighting [GH06], ambient occlusion [McG10], and precomputed radiance transfer (PRT) [SKS02]. We apply our method to these applications and present an additional optimization for diffuse BRDFs. While our approach scales well in the number of integration samples (or, put differently, the complexity of the function being integrated), it does not scale as well as ray-tracing with the size of the scene. As such, we are competitive with optimized ray-tracing on scenes of low-to-moderate complexity.

2. Related Work

We address a general problem with immediate applications in rendering and discuss previous work in related areas.

Boundaries and Sampling. Silhouette computation is critical to many non-photorealistic approaches [DFRS03] and we are motivated by the large literature in this area. Hertzmann and Zorin [HZ00] use an octree defined over a 4D dual mesh to quickly find contour edges, whereas we employ a bounding volume hierarchy (BVH). Silhouette extraction using Hough transforms may accelerate our approach [OZ06].

Shadow volumes [Cro77] are encased by planes defined by object silhouette edges seen from a point-light’s position. Depth counting from the eye determines whether a pixel is in shadow or not. Before we can consider integrating visibility-masked functions, we must compute *spherical* silhouettes at all shade points, which is significantly more difficult.

Chen and Arvo [CA00] derive analytic expressions for irradiance from polygonal lights and reduce the irradiance computation to a contour integral over the edges of the polygonal source using Stokes’ theorem. Similarly, McGuire approximates ambient occlusion (AO), using Stokes’ theorem

to integrate over the edges of an extruded *ambient occlusion volume* [McG10]. We also integrate over boundaries, but to compute accurate integrals of visibility-masked functions.

Laine et al. [LAA*05] and Lehtinen et al. [LLA06] combine ray-tracing with a silhouette based approach to compute direct illumination from planar polygonal light sources. We also exploit silhouettes to simplify the representation of visibility, although we consider the more general problem of a textured spherical light domain. We additionally use line sampled integration, whereas Laine et al. and Lehtinen et al. sample with points. Many additional works leverage similar silhouette and depth complexity sampling in order to efficiently compute soft shadows [FBP08, AAM03, ED07], but none consider semi-analytic spherical integration. More recent works leverage lazy visibility evaluation and analytic from-polygon form-factor computation to compute soft shadows from geometric area lights and AO effects [MAAG12, AMGA12]. We are motivated by works on semi-analytic visibility [NN85, SR00, HMN05] which compromise between stochastic sampling approaches that are sensitive to noise, and more complex fully analytic solutions.

By determining the *spherical visibility*, we can compute integrals of arbitrary visibility-masked functions. We discuss the differences between ours and the aforementioned approaches in Section 3 but, put briefly, we generalize prior approaches to the spherical domain while leveraging line, instead of point, sampling for semi-analytic integration.

Many recent approaches use line samples to perform numerical integration more efficiently than with point samples. Gribel et al. [GBAM11] use line samples to compute motion blurred ambient occlusion, and Barringer et al. [BGAM12] determine camera visibility for geometric curve primitives (e.g., hair) using per-pixel line sampled integration. These techniques do not use silhouettes when testing visibility and consider different visibility scenarios (e.g., not general environment map occlusion). The way we perform line integration is similar to the evaluation of invisibility proposed by Appel [App67], as well as drawing parallels to fundamental work in prefiltering and anti-aliasing [GT96, JP00].

Wang et al. [WRG*09] use a visibility representation, suit-

able for spatial interpolation, based on signed-distance functions. They still use point-sampling to project visibility into a basis-space representation for relighting: this projection can be accelerated with our approach (see Section 4). Annen et al. [ADM*08] sample environment maps at few (e.g., 30) directions and use low-resolution filtered shadow maps to sample visibility for all-frequency shadows of dynamic scenes. Their filtering masks typical spatial and angular shadow mapping bias at the cost of blurring the shadows. We instead determine spherical visibility geometrically and accurately compute semi-analytic integrals of visibility-masked lighting and BRDFs without any noise or blurring.

PRT and Analytic Rendering Approaches. Recent work focused on static geometry- and image-relighting using spherical harmonics (SH) [SKS02], Haar Wavelets [NRH03, NRH04], or radial basis functions [TS06, WRG*09]. These approaches represent the lighting, BRDF, visibility, or a combination of these terms in basis expansions, and then perform relighting entirely in the basis-space.

Some PRT-based approaches approximate visibility at runtime to handle dynamic geometry by rasterizing blockers into a hemisphere [KLA04], multiplying precomputed volumetric visibility for rigid blockers using basis-space product operators [ZHL*05], or using simplified blocker proxies and accumulating logarithmic visibility in the basis-space [RWS*06]. GPU accelerated non-linear projections [GHFP08] could be employed in the context of sampling per-pixel spherical visibility in parallel, however these techniques scale linearly in the size of the scene.

Ramamoorthi and Hanrahan [RH01, RH02] show that outgoing radiance is a convolution of the (radially-symmetric) view-evaluated BRDF and incident radiance. By representing BRDF and lighting in SH, this convolution simplifies to a frequency-space product. In the case of diffuse reflection, they show that an order-3 SH expansion of the *clamped-cosine* ensures a maximum shading error $\leq 3\%$. They compute unshadowed shading, where incident radiance is equal to the environment light (no visibility). Sloan [Slo08] shows that even this slight error can become noticeable with high dynamic range (HDR) lighting. With our semi-analytic integration, *shadowed* incident radiance can be computed and used in a frequency-space shading context. In fact, we will show that when clamping incident light to the upper-hemisphere, diffuse shading can be *perfectly* computed with *all-frequency* shadows using only *band-1* SH (3 coefficients instead of 9). This resembles a vector irradiance formulation by Arvo [Arv95] (see Section 4 and Appendix A).

3. Our Method

We integrate spherical functions $f(\omega)$ masked by visibility:

$$I = \int_{S^2} V(\omega) f(\omega) d\omega, \quad (1)$$

where S^2 are all unit directions and f may be scalar or vector valued. Since visibility is binary, we can write:

$$I = \int_{\{\omega | V(\omega)=1\}} f(\omega) d\omega. \quad (2)$$

To compute the integral over the unoccluded region, $\{\omega | V(\omega) = 1\}$, we need to find the silhouette of all the occluders in the scene. Previous approaches which consider visibility silhouettes [LAA*05, LLA06] still require ray-tracing and only compute shading from polygonal lights. We instead consider the full spherical visibility (silhouette) and compute arbitrary integrals of the form in Equation 1. Finding only the **silhouette** edges is a difficult problem because whether an edge is a silhouette edge or not may depend on occluders far from the edge. Our approach is instead to find the **contour** edges: edges which contain the silhouette and can be quickly found using a dual-space BVH (see Section 3.1.)

Given this superset of edges, we could use 2D geometric booleans to determine which ones comprise the silhouette, but this is a costly and error-prone procedure. Instead, we reduce the dimensionality of the problem by parameterizing the sphere by u and v (we discuss our specific choice of parameterization in Section 3.3), rewriting Equation 2 as

$$I = \int_u \left(\int_{\{v | V(u,v)=1\}} f(u,v) J(u,v) dv \right) du, \quad (3)$$

where $J(u,v) = \left\| \frac{d\omega}{du} \times \frac{d\omega}{dv} \right\|$. For each u , the region $\{v | V = 1\}$ is a set of disjoint intervals, allowing us to write:

$$I = \int_u \sum_i^{N(u)} \left(\int_{v_i^-(u)}^{v_i^+(u)} f(u,v) J(u,v) dv \right) du, \quad (4)$$

where $N(u)$ is the number of intervals and $[v_i^-(u), v_i^+(u)]$ is the i^{th} interval. Finally, we discretize along the u dimension, decomposing the 2D problem into a set of 1D problems:

$$I = \sum_u \sum_i^{N_u} \left(\int_{v_{i,u}^-}^{v_{i,u}^+} f(u,v) J(u,v) dv \right) \Delta u, \quad (5)$$

where N_u , $v_{i,u}^-$, and $v_{i,u}^+$ are the discrete analogues of the continuous variables $N(u)$, $v_i^-(u)$, and $v_i^+(u)$. In other words, we need only find the intersection points of the silhouette with u -isolines, as illustrated in Figure 3. The depth complexity function (DCF; Section 3.1) allows us to find these points efficiently. This basic pattern of boolean dimensionality reduction dates back to the early days of ray casting (e.g., Roth [Rot82]); we use it for finding integration bounds. The way we evaluate the definite integral $\int_{v_{i,u}^-}^{v_{i,u}^+} f(u,v) J(u,v) dv$ depends on f ; we discuss specifics in Section 3.2.

3.1. Contour Edge Computation

At a point \mathbf{p} , we are interested in the visibility function $V(\omega)$, which specifies whether a ray starting at \mathbf{p} towards ω intersects any scene geometry. Although this definition suggests

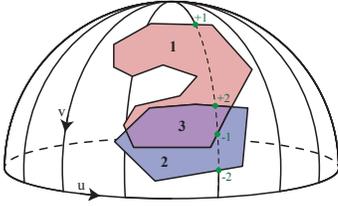


Figure 3: Visibility diagram of two polyhedral occluders projected onto the sphere. W values (Section 3.1) are shown as well as unoccluded portions of u -isolines. The DCF increments along a u -isoline are shown in green. Examples of this picture using our parameterization are in Figure 5.

ray-tracing, we entirely avoid ray-tracing in our method. The visibility function is piecewise-constant (in fact, binary) and it changes precisely when the ray crosses silhouette edges, by definition. To find the silhouette edges, we define the DCF, $W : S^2 \rightarrow \mathbb{N}_0$, which generalizes V .

For a direction ω , $W(\omega)$ counts the times a ray starting at \mathbf{p} in that direction intersects the scene geometry. This function is also piecewise constant and its value changes when the ray crosses *contour edges*, as shown in Figure 3. Although the DCF is more complicated than visibility, silhouettes are defined by a global condition, whereas contour edges are defined locally, making them easier to compute.

We assume our scenes are represented with triangle meshes. Consider what happens to $W(\omega)$ at a fixed \mathbf{p} as we vary ω so that the ray crosses a triangle edge e . If e is a boundary edge, i.e., has only one adjacent triangle, then the ray goes from not intersecting this triangle to intersecting it or vice versa. In this case $W(\omega)$ changes by ± 1 . If e is adjacent to two triangles, there are two possibilities: the ray can go from intersecting one of the triangles to intersecting the other, or the ray can go from intersecting neither to intersecting both (Figure 4). Assuming triangles are oriented consistently, if the triangles are both front-facing w.r.t \mathbf{p} or both back-facing w.r.t \mathbf{p} , then the first possibility occurs and $W(\omega)$ does not change. If, on the other hand, one of the triangles is front-facing and the other is back-facing (w.r.t \mathbf{p}), then $W(\omega)$ changes by ± 2 when the edge is crossed.

Let \mathbf{f}_1 and \mathbf{f}_2 be plane equations of the triangles, expressed as 4D vectors (the first three components are the normal, the fourth is the offset). Then W changes, according to the conditions detailed above, when an edge is crossed and iff:

$$\text{sign}(\mathbf{f}_1 \cdot \mathbf{p}) \neq \text{sign}(\mathbf{f}_2 \cdot \mathbf{p}), \quad (6)$$

where \mathbf{p} is expressed in homogeneous coordinates. The contour edges for shade point \mathbf{p} are therefore the mesh boundary edges and edges that satisfy Equation 6.

Finding the mesh boundary contour edges, specifically the mesh edges with only one adjacent triangle, is simple since they are the same at every shade point. To quickly find the

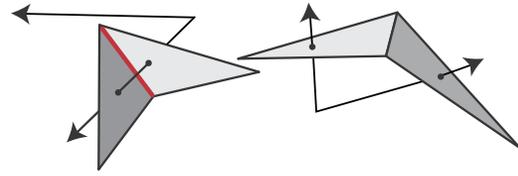


Figure 4: Left: W changes by 2 when the edge is crossed (making it a contour). Right: no change in W (not a contour).

non-boundary contour edges, we use a method similar to that of Hertzmann and Zorin [HZ00], discussed below.

For scenes with smooth objects, only a small fraction of edges are contour edges. To avoid testing whether each edge satisfies Equation 6, we represent each edge e (in the primal domain) as a line segment in a dual 4D space with endpoints \mathbf{f}_1 and \mathbf{f}_2 . These 4D line segments form a 4D mesh in the dual space which we place into a 4D BVH. To find contour edges, we perform a 3D-hyperplane (corresponding to the normal at \mathbf{p} in the dual space) vs. 4D dual mesh intersection using the 4D BVH. We are only interested in the visible half-space $(\mathbf{x} - \mathbf{p}) \cdot \mathbf{n} \geq 0$ and we filter out contour edges having both endpoints in the invisible half-space. Since subsequent processing is done on a sphere, we project the contour edges onto the unit sphere around \mathbf{p} , making each edge the arc of a great circle. At a high level, the above resembles the way Laine et al. [LAA*05] consider silhouettes; however, we compute integrals over *line* samples and handle arbitrarily large domains. We discuss these and other distinguishing differences below.

3.2. Computing the Final Integral

We now use the contour edges to evaluate Equation 5 by computing the depth complexity W along every discrete u -isoline and evaluating the innermost integral along segments where W is zero. By construction, W changes at intersections of the contour edges and the u -isolines. We therefore associate a bucket with each u -isoline where we store these intersections. For each edge, we determine which discrete u -isolines it intersects, and put the v values of the intersections as well as the change of W (± 1 or ± 2) into the associated buckets. We then sort each bucket, to obtain the changes to W in order of increasing v . For a u -isoline, suppose we know the value of W at $v = 0$. By traversing the sorted bucket, we can then incrementally compute the value of W for each v and integrate $f(u, v) J(u, v)$ over the segments where it is 0.

There are two important caveats. First, determining the value of W at $v = 0$. By choosing a parameterization so that all u -isolines start at the same spherical point, we reduce the problem to computing W at that single point. The robust solution is to trace a ray in that direction [LAA*05]; we implement a simpler solution: we pick an arbitrary W starting value, traverse the buckets to determine the minimum W and then offset the starting value so that this minimum value is zero. The

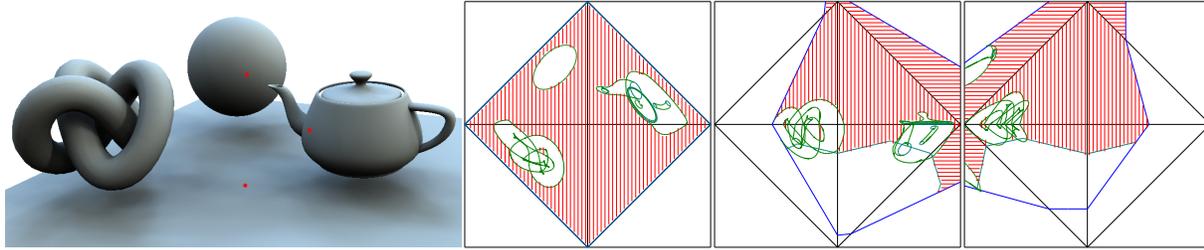


Figure 5: Three unfolded octahedra show the visibility about the three red points in the scene, in order of the point on the plane, on the sphere, and on the teapot. Octahedral edges are black, the visible half-space boundary is blue, contour edges are green, and unoccluded sections of u -isolines are red. y and z axes are reversed for the sphere and teapot points (see Section 3.3).

second caveat is that we filter out contour edges in the invisible half-space, so if a u -isoline crosses into this half-space and then crosses back, the values for W will not be correct. We choose the parameterization so that once u -isolines cross into the invisible half-space, they never cross back.

Although Laine et al. [LAA*05] use silhouettes to represent visibility, in addition to differences in silhouette detection and applications, our integration method is itself fundamentally different. In their method, light integration is based on planar point samples, while we integrate semi-analytically along entire contiguous spherical (iso)lines. As a result, our integration scales as $O(\sqrt{k})$ in the number of effective samples k , while the integration in their method scales as $O(k)$.

3.3. Spherical Parameterization

The simplest parameterization to use is spherical angles. The u and v coordinates are azimuthal and zenith angles. All u -isolines start at the north pole and, as long as the north pole is in the visible half-space, u -isolines that cross into the invisible half-space never cross back. If the north pole is not in the visible half-space, we simply reverse the direction of u and start from the south pole. Unfortunately, computing the v coordinates of edge intersections with u -isolines requires evaluating trigonometric functions, which is expensive.

Our implementation instead uses an octahedral parameterization, similar to Praun and Hoppe [PH03]. Instead of projecting edges onto the unit sphere, we project them through the origin onto a unit octahedron $|x| + |y| + |z| = 1$ (Figures 5

and 6). This projection maps (x, y, z) to $\frac{(x, y, z)}{|x| + |y| + |z|}$ and has the property that, within an octant, straight lines map to straight lines. This implies that for an edge within an octant, the v coordinate of the intersection of the u -isolines with the projection of that edge is a simple linear function of u and does not require trigonometry to evaluate. We use the x and y coordinates of our projection as u and v . The z coordinate is then determined up to sign: we have a set of buckets for $+z$ and a set of buckets for $-z$. This property allows our intersections to be computed roughly twice as fast as for the spherical parameterization. The change-of-variables term of this parameterization is $J(u, v) = (u^2 + v^2 + (1 - |u| - |v|)^2)^{1.5}$.

The situation with our two caveats from Section 3.2 is somewhat more complicated in the octahedral parameterization. First, when determining the value of W at $v = 0$, the u -isolines all start at different points which requires us to calculate W values at every point. We do this by recording all projected edge intersections with $v = 0$ into a special bucket. We can then calculate W at $\omega(u, 0)$ incrementally from $W(\omega(0, 0))$ by traversing the elements in this bucket. The second issue is that in this parameterization, a u -isoline can cross into the invisible half-space and then back out. However, this can only happen if $|n_y| < |n_z|$ and in this case we simply reverse the y and z axes, temporarily rotating the domain in order to avoid the double-crossing event.

4. Applications

We focus on applications in realistic rendering, where integrals of the form in Equation 1 commonly arise. Each application can be fully described by its $f(\omega)$. Our algorithm can sample shade points arbitrarily; we use per-pixel shading in all cases except for per-vertex PRT vector computation.

For a function $f(\omega)$ (e.g. environment map) we compute a summed area table (SAT) over u -isolines, reducing definite integral computation to two lookups and a subtraction. Evaluating f only affects SAT construction cost, which is negligible compared to the cost of silhouette search and integration. We discuss below how basis projections can be combined with our SAT to handle higher-dimensional functions.

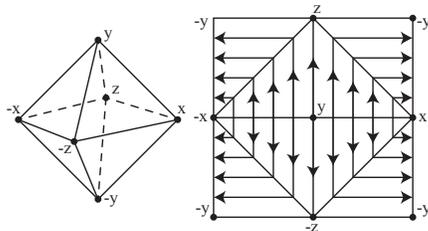


Figure 6: Unfolded octahedron (left) and u -isolines (right).

Ambient Occlusion. We compute AO with $f(\omega) = [\mathbf{n} \cdot \omega]_0$. Note that AO sometimes refers to average visibility [LS10] without the cosine; this value corresponds to $f = 1$, assuming no AO fall-off functions are employed. Our approach cannot support typical distance-to-intersection AO fall-off functions since we only mask functions according to binary visibility. Figures 9 and 11 illustrate this application.

Precomputed Radiance Transfer. We apply our approach to precomputing basis projections of visibility in PRT triple-product relighting. In this case, $f(\omega) = \mathbf{B}(\omega)$, where \mathbf{B} is a vector-valued function of the basis functions. We use the real-valued SH basis functions, $\mathbf{B} = \mathbf{y}$ where

$$y_l^m(\omega) = \begin{cases} K_l^0 P_l^0(\cos \omega_\theta), & m = 0 \\ \sqrt{2} K_l^m \cos(m\omega_\phi) P_l^m(\cos \omega_\theta), & m > 0 \text{ and} \\ \sqrt{2} K_l^{|m|} \sin(|m|\omega_\phi) P_l^{|m|}(\cos \omega_\theta), & m < 0 \end{cases}$$

$$\mathbf{y} = [y_0^0(\omega), y_1^{-1}(\omega), y_1^0(\omega), y_1^1(\omega), y_2^{-2}(\omega) \dots] \quad (7)$$

and m indexes the $(2l + 1)$ band- l basis functions, K_l^m is a normalization term, and P_l^m are associated Legendre polynomials. Note however that any basis with a double- or triple-product formulation can be employed, including data-driven bases [NKLN10]. By multiplying \mathbf{B} with an arbitrary f , we can also project visibility-masked functions onto a basis which is different, and generally more accurate, than performing a product with visibility in the basis space.

Image relighting. For a fixed view, we can precompute silhouettes and modify the light at run-time. Therefore, we integrate the product of visibility and view-evaluated BRDF, $f_r(\omega, \omega_o)$, weighted by cosine foreshortening: $f(\omega) = f_r(\omega, \omega_o) [\mathbf{n} \cdot \omega]_0$, where ω_o is the view vector.

All-Frequency Diffuse Shading. Diffuse shading is the convolution of a clamped cosine kernel, aligned along \mathbf{n} , with the incident radiance. Since this kernel is clamped to the upper hemisphere, the sharp discontinuity along the equator in the angular domain causes infinite frequencies in its SH projection. Ramamoorthi and Hanrahan [RH01] show that a 9 coefficient, order-3 SH projection of this kernel captures the majority of its energy, despite its infinite frequency content. They project the unshadowed incident radiance (the environment map, for distant lighting) into order-3 SH and shade using analytic SH convolution.

We will show that, by incorporating hemispherical clamping into the incident radiance computation *before* projecting into SH, we can compute all-frequency shadows with a band-limited *unclamped* cosine, requiring only 3 SH coefficients. We avoid soft-shadowing approximation errors caused by SH projection in PRT by projecting *visibility-masked* incident illumination. We discuss below how to perform final shading using a fully band-limited BRDF kernel.

Diffuse direct lighting is computed as the following integral:

$$I = \frac{\rho}{\pi} \int_{S^2} L_{in}(\omega) V(\omega) H(\omega, \mathbf{n}) (\mathbf{n} \cdot \omega) d\omega, \quad (8)$$

where the hemispherical clamping function, $H(\omega, \mathbf{n})$, is 1 when $\mathbf{n} \cdot \omega > 0$ and 0 otherwise. The integral of a product of two spherical functions is the dot-product of their SH projection coefficient vectors. Previous approaches projected the functions $L_{in}(\omega)$ and $V(\omega)H(\omega, \mathbf{n})(\mathbf{n} \cdot \omega)$ into SH, because this enabled run-time manipulation of lighting for static geometry. Both of these terms have unbounded frequency content, which means that the accurate computation of the dot-product of their coefficients requires infinitely many terms. Ramamoorthi and Hanrahan's observation relates to the fact that, while the clamped cosine has unbounded frequency content, most of it is concentrated in the first three SH bands.

In our case, to avoid precomputing a SAT for each normal, we need to decouple incident light from $\mathbf{n} \cdot \omega$ by integrating the hemispherical clamping in our silhouette integration of L_{in} , treating it as a part of visibility. The incorporation of hemispherical clamping in the silhouette computation is performed by default, as discussed in Section 3.1.

The remaining function, $\mathbf{n} \cdot \omega$, is *unclamped* and has *bounded* frequency content. This function only has non-zero SH projection coefficients in the linear $l = 1$ band. This permits us to compute the integral of the product of these two terms using a single SH band, meaning we need only retain the linear band of the SH projection of $L_{in}(\omega) H(\omega, \mathbf{n})$.

We obtain these three coefficient with $f(\omega) = L_{in}(\omega) H(\omega, \mathbf{n}) \mathbf{y}_1(\omega)$, where $\mathbf{y}_1(\omega) = [y_1^{-1}(\omega), y_1^0(\omega), y_1^1(\omega)]$ is a vector valued function of the three linear SH basis functions (Appendix A relates this discussion to vector irradiance). This representation of incident radiance can be extended to non-diffuse BRDFs where, unlike prior work, shading can be computed more accurately and efficiently.

Glossy Rendering. We compute the optimal SH projection of incident radiance depending on the BRDF. For diffuse BRDFs, we only use $l = 1$, whereas for Phong BRDFs we choose the SH order according to guidelines set by Ramamoorthi and Hanrahan [RH02]: namely, that a larger SH expansion is required as the Phong exponent increases. As opposed to shading uniformly with a maximum SH order regardless of the underlying BRDF, this approach guarantees that additional computation is only performed as needed. We do, however, precompute a single SAT for a maximum SH order. Figures 1 and 9 illustrate scenes with both diffuse and glossy BRDFs. While we chose SH to leverage our new diffuse formulation, other basis representations could easily be substituted (i.e. different families of $\mathbf{B}(\omega)$).

Material Editing. We can support a simple material editing scenario: in the context of image-relighting, we allow a user to interactively change the BRDF of the objects in a

Scene	# Triangles	Env.	Ours	QMC	Ours	QMC	Ours	QMC
			(500 ²)	(75 ² –85 ²)	(50 ²)	(50 ²)	(10 ²)	(10 ²)
BMW	50.2k	Ambient	1m:24s	6m:37s	1m:10s	3m:42s	0m:56s	0m:28s
Fly 1	89.9k	Uffizi	4m:07s	8m:16s	3m:27s	3m:57s	2m:55s	0m:21s
Fly 2	89.9k	Grove	3m:19s	4m:20s	2m:49s	2m:35s	2m:16s	0m:23s
Rhino Car	123.2k	Grace	6m:16s	4m:22s	5m:47s	1m:32s	4m:53s	0m:27s
Teapot Grid	1.9M	Grove	31m:57s	17m:33s	30m:12s	8m:03s	28m:22s	1m:18s

Table 1: Timing comparison of our approach and importance-sampled QMC for different sampling rates (in brackets). With few samples QMC outperforms our technique and scales well with scene complexity. Our approach can outperform QMC at sample rates necessary to obtain artifact-free images for our scenes with low-to-mid geometric complexity (see Figure 10).

scene. By computing the basis projection of cosine-weighted incident radiance with $f(\omega) = L_{in}(\mathbf{p}, \omega) [\mathbf{n} \cdot \omega]_0 \mathbf{y}(\omega)$ using a pre-determined number of basis functions (unlike the diffuse case), we can compute the basis projection of $f_r(\omega, \omega_o)$ on-the-fly and shading is computed with a dot-product of the projection coefficients [SKS02] (see Figure 7).



Figure 7: Editing the car’s paint in Figure 1 from glossy to diffuse, and changing its texture. The BRDF is incorporated at shade time, making material changes instantaneous.

5. Results and Discussion

Our experiments were performed on a 4-core hyper-threaded Intel Core i7-2600K, with an NVidia GTX 560. In our method, the GPU was only used for final shading; the actual integral computation was done in parallel on the CPU, both for our method and for Monte Carlo. Because we do not exchange data between shade points, our memory usage beyond the BVH and the output texture is negligible: even our largest scene used less than 170 MB of RAM.

We use Intel’s optimized Embree v1.1 [Int11] ray-tracer for comparisons but, like our method, the surrounding code (ray generation, sampling, shading, etc.) is not as optimized as the ray-tracing engine. Embree is optimized for tracing incoherent rays, and we note that our AO and/or direct-illumination computations generate more coherent ray structures. Once Embree constructs the ray-tracing acceleration structure (a BVH variant), we compute a PDF according to the environment light then generate and warp deterministic Halton distributions by this PDF, at each shade point. At shading time, we reject (importance-sampled) directions in the lower hemisphere and launch shadow rays for the remaining samples, evaluating the BRDF at these directions.

The performance of our integration is largely independent of $f(\omega)$ ’s complexity, and is primarily a function of scene

complexity. The majority of our algorithm’s processing time is spent finding visibility silhouettes; once found, these are used during final integration which scales linearly with \sqrt{k} but only composes a small portion of the final render time. The cost to construct the BVH and the SAT for f is negligible compared to the cost of integrating at all the points. The cost of integrating at a point is roughly proportional to the number of contour edges that point sees (subject to the cost of finding them using the BVH). For a scene with n triangles, the number of contour edges for an average shade point increases roughly in proportion to $n^{0.8}$ [McG04].

Although asymptotic analysis ignores many considerations, it can nevertheless shed light on how different algorithms scale. If we use m shade points ($m \leq 1024 \times 768$ in our examples) and a sampling rate of k for f , our runtime scales roughly as $O(m\sqrt{kn}^{0.8})$. Due to the 4D BVH, extracting the contour edges is roughly proportional to their number $O(mn^{0.8})$ but with a larger constant. Our method takes approximately the same time to compute ambient occlusion, all-frequency direct lighting, and PRT vectors. In contrast, Monte Carlo integration is dominated by ray-tracing and shading. Tracing a ray has approximately logarithmic complexity in n (the number of triangles) therefore, for similar sampling rates, Monte Carlo will require roughly $O(mk \log n)$ time. The actual sampling rate depends on the lighting: as Figure 11 shows, Monte Carlo requires more samples for low-frequency lighting, as importance sampling cannot adapt to the variance. Similarly, Monte Carlo requires more rays to render glossy surfaces: glossy BRDFs require different sampling distributions at every shade point, reducing the effectiveness of light-importance sampling. This can be alleviated to a certain degree with BRDF-sampling in a multiple importance sampling framework [VG95]. For larger scenes, ray-tracing overtakes our method (Figure 12).

“Samples” in our approach correspond to the number of uv iso-buckets into which we semi-analytically discretize the visibility silhouette for integration; our sampling resolution is set to match $f(\omega)$ ’s SAT resolution. For QMC, “samples” correspond to the number of integration rays launched at each pixel. We use independent sampling sets at each pixel for QMC, resulting in noise when features are under-

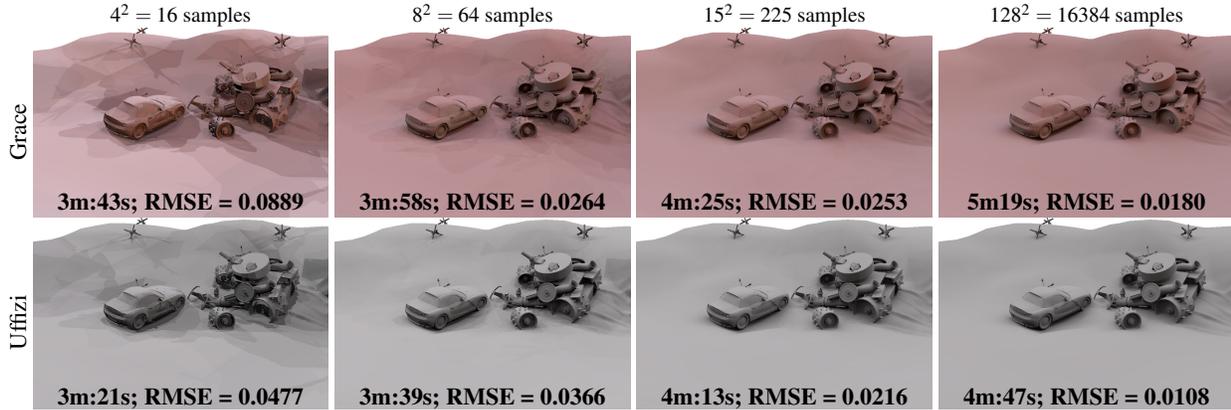


Figure 8: Convergence with increasing sample counts on the Tank Car scene (85.2k triangles). Banding is caused by the uniform discretization of $f(\omega)$ in our SAT shared across pixels. Our approach always converges to ground truth with $k = 10^6$.

sampled, whereas a global coordinate system is used when bucketing silhouette segments in our approach, resulting in banding artifacts for under-sampled features. For low-to-medium geometric complexity and at the higher sampling rates required to obtain converged results, our approach occasionally outperforms QMC (see Figures 10 and 11).

When integrating higher-dimensional (e.g. BRDFs) or spatially-varying functions, basis projections can be employed and a component of our render time will contain a linear dependence on the size of the basis expansion. At every shade point, we cannot easily consider only a subset of the scene geometry when computing silhouettes; as such, we do not support AO with arbitrary fall-off functions. Our approach is consistent and, for high-frequency environment maps, a sampling rate of 64^2 to 256^2 ($k = \{64^2, 256^2\}$) is sufficient to generate converged results. The time required to compute a single spherical integral with our algorithm grows much more rapidly with scene complexity than with sampling rate. For simpler scenes (Figure 2 and BMW), this time varies roughly between 0.06ms and 0.13ms. For moderately complex scenes (Fly 1, Fly 2, Rhino Car, Tank Car) it varies roughly between 0.26ms and 0.61ms. The time per integral for Crash (184.5k triangles; Figure 11) and Teapot Grid (1.9M triangles; Figure 12) were from 0.79ms to 2.48ms.

Timings are reported in Table 1, Figures 8 and 10. Images are rendered at 1024×768 and equal-quality comparisons use an $RMSE \approx 0.01$ vs. ground truth computed with QMC and $k = 10^6$: from Figures 8 and 10 we note that at sample counts over $45^2 = 2025$ the majority of the QMC images visually converge to almost noise-free results. QMC typically exhibits better performance vs. quality behavior than our approach at low sample counts and scales much better with geometric complexity; however, our approach scales better with the sampling rate ($O(\sqrt{k})$ vs. $O(k)$ for QMC).

Figure 10 compares our convergence/performance profile to QMC. At lower sampling rates, QMC has a favorable performance vs. error behavior. However, at the high sampling

rates required to obtain converged results on scenes like ours with low-to-moderate geometric complexity, we can outperform the QMC solution (see e.g. BMW, Fly 1 and Fly 2 with $k = 45^2$). The convergence of our approach (as a function of k) depends on $f(\omega)$ as e.g. higher-frequency lighting environments may exhibit more drastic banding artifacts during convergence (Figure 8); however, in all cases we observed visual convergence with $64 \leq \sqrt{k} \leq 256$. Figure 9 illustrates our approach on scenes with progressively higher-frequency lighting, and we stress-test our approach on a scene with 1.9 million triangles (Figure 12). While our approach does not scale as favorably with scene complexity as ray-tracing, which benefits from three decades of research on acceleration structures, we hope that our work will motivate investigations on such structures for silhouette extraction.

6. Future Work

Our algorithm is data-parallel, implemented on a multi-threaded CPU, but a high-end 4-core CPU is not capable of as many FLOPS as a single GPU. We leave GPU acceleration of our approach to future work. Our approach is orthogonal to adaptive spatial and image-space sampling (e.g., Sloan et al. [SGNS07]), and our approach can be extended to compute occlusion-aware irradiance gradients [RMB07] for irradiance caching [WRC88] with environment lighting.

A preliminary investigation of building a hierarchy over shade points using a Cartesian product tree, similar to hierarchical penumbra casting [LA05], only yielded marginal performance improvements however this and other adaptive approaches are interesting directions of future work. Extensions to indirect lighting and dynamic geometry are left for future work. The latter will benefit from research in fast BVH reconstruction (we currently use a simple axis-aligned bounding box BVH). Our approach does not scale well to complex geometry, e.g. foliage, especially when this complexity increases the DCF. Addressing these shortcomings mandates more scalable silhouette searching.

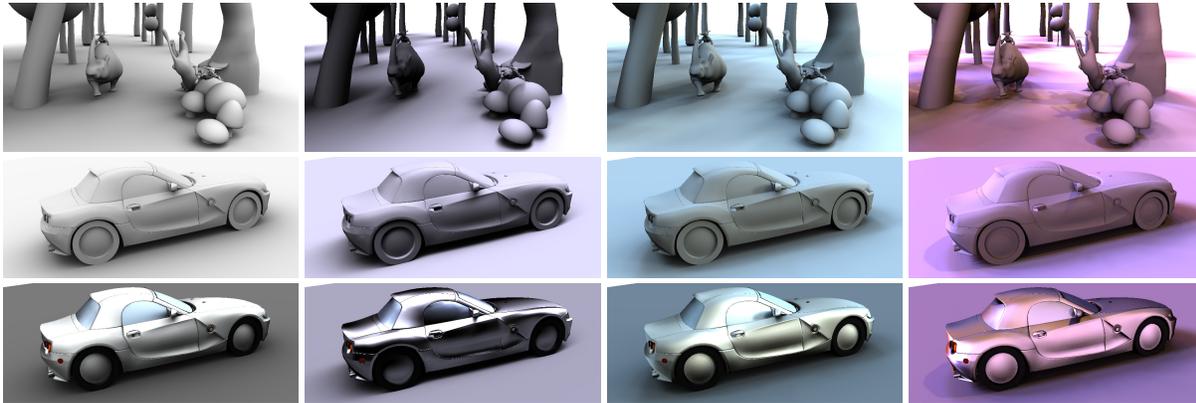


Figure 9: Rendering with different lighting: (left to right) ambient, Uffizi, Grove, and Grace. Render times for each scene (Fly 1 and BMW) are independent of lighting and take (top to bottom) 4m:07s, 1m:24s, and 1m:29s. Changing the materials from diffuse to glossy in the BMW scene does not add significant overhead, due to our frequency-space optimization.

7. Conclusion

We have presented a semi-analytic approach for computing spherical integrals of visibility-masked functions. We demonstrate the feasibility of our approach on several rendering applications and, while our approach does not scale as well as Monte Carlo ray-tracing on complex scenes, our proof of concept competes with optimized ray-tracing on scenes with low-to-moderate geometric complexity. Our algorithm is data-parallel, has a modest memory footprint, and its performance is independent of the function being integrated (depending only on the function’s resolution).

We hope our approach will promote work on new acceleration structures for silhouette detection to improve the performance and scalability of our approach on complex scenes.

References

- [AAM03] ASSARSSON U., AKENINE-MÖLLER T.: A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics* 22, 3 (July 2003), 511–520. [2](#)
- [ADM*08] ANNET T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph.* 27, 3 (2008), 1–8. [3](#)
- [AMGA12] APOSTU O., MORA F., GHAZANFARPOUR D., AVENEAU L.: Analytic ambient occlusion using exact from-polygon visibility. *Computers & Graphics* 36, 6 (2012). [2](#)
- [App67] APPEL A.: The notion of quantitative invisibility and the machine rendering of solids. In *Proceedings of the 22nd national conference* (NY, USA, 1967), ACM, pp. 387–393. [2](#)
- [Arv95] ARVO J.: *Analytic Methods for Simulated Light Transport*. PhD thesis, Yale University, Dec. 1995. [3](#), [12](#)
- [BGAM12] BARRINGER R., GRIBEL C. J., AKENINE-MÖLLER T.: High-quality curve rendering using line sampled visibility. *ACM Trans. Graph.* (2012). [2](#)
- [CA00] CHEN M., ARVO J.: A closed-form solution for the irradiance due to linearly-varying luminaires. In *Proceedings of Rendering Techniques* (2000). [2](#)
- [Cro77] CROW F. C.: Shadow algorithms for computer graphics. *Proceedings of SIGGRAPH 11*, 2 (July 1977). [2](#)
- [DDP02] DURAND F., DRETTAKIS G., PUECH C.: The 3d visibility complex. *ACM Trans. Graph.* 21, 2 (2002). [1](#)
- [DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Transactions on Graphics (SIGGRAPH)* 22, 3 (July 2003). [2](#)
- [ED07] EISEMANN E., DÉCORET X.: Visibility sampling on GPU and applications. *Computer Graphics Forum* (2007). [2](#)
- [ESAW11] EISEMANN E., SCHWARZ M., ASSARSSON U., WIMMER M.: *Real-Time Shadows*. A.K. Peters, 2011. [1](#)
- [FBP08] FOREST V., BARTHE L., PAULIN M.: Accurate shadows by depth complexity sampling. *Computer Graphics Forum* (2008). [2](#)
- [GBAM11] GRIBEL C. J., BARRINGER R., AKENINE-MÖLLER T.: High-Quality Spatio-Temporal Rendering using Semi-Analytical Visibility. *Transactions on Graphics*, (2011). [2](#)
- [GH06] GHOSH A., HEIDRICH W.: Correlated visibility sampling for direct illumination. *Visual Computer* 22 (2006). [2](#)
- [GHFP08] GASCUEL J.-D., HOLZSCHUCH N., FOURNIER G., PEROCHE B.: Fast Non-Linear Projections using Graphics Hardware. In *ACM Symposium on Interactive 3D Graphics and Games* (feb 2008). [3](#)
- [GT96] GUENTER B., TUMBLIN J.: Quadrature prefiltering for high quality antialiasing. *ACM Transactions on Graphics* 15, 4 (Oct. 1996), 332–353. [2](#)
- [HMN05] HAUMONT D., MÄKINEN O., NIRENSTEIN S.: A low dimensional framework for exact polygon-to-polygon occlusion queries. In *Proceedings of the Conference on Rendering Techniques* (2005), EGSR’05, Eurographics, pp. 211–222. [2](#)
- [HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *Proceedings of SIGGRAPH* (NY, 2000), ACM. [2](#), [4](#)
- [Int11] INTEL CORP.: Embree, 2011. software.intel.com/en-us/articles/embree-photo-realistic-ray-tracing-kernels/. [7](#)
- [JP00] JONES T. R., PERRY R. N.: Antialiasing with line samples. In *Proceedings of the Eurographics Workshop on Rendering Techniques* (June 2000), pp. 197–206. [2](#)
- [KLA04] KAUTZ J., LEHTINEN J., AILA T.: Hemispherical rasterization for self-shadowing of dynamic objects. In *Rendering Techniques* (2004). [3](#)

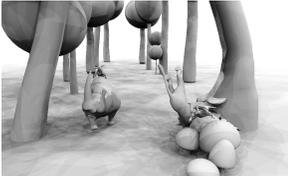
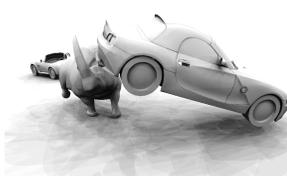
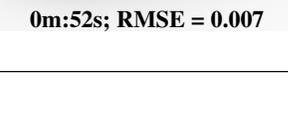
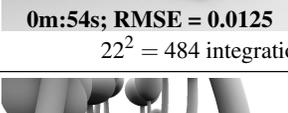
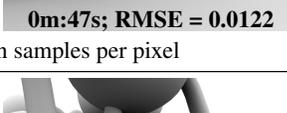
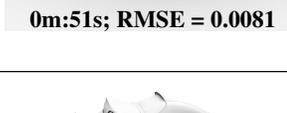
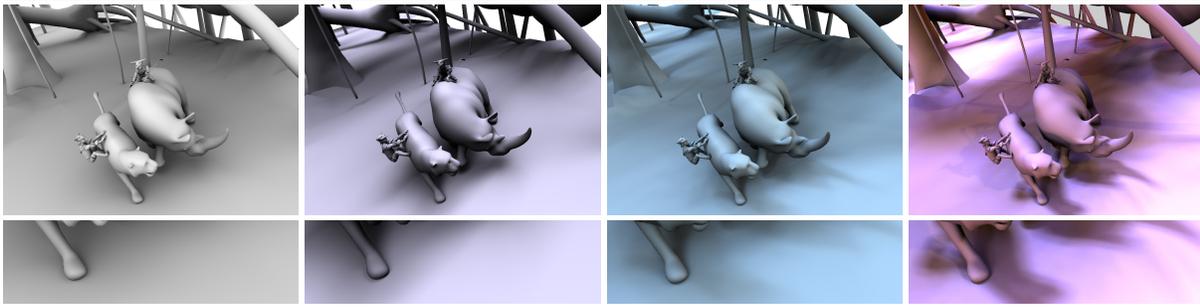
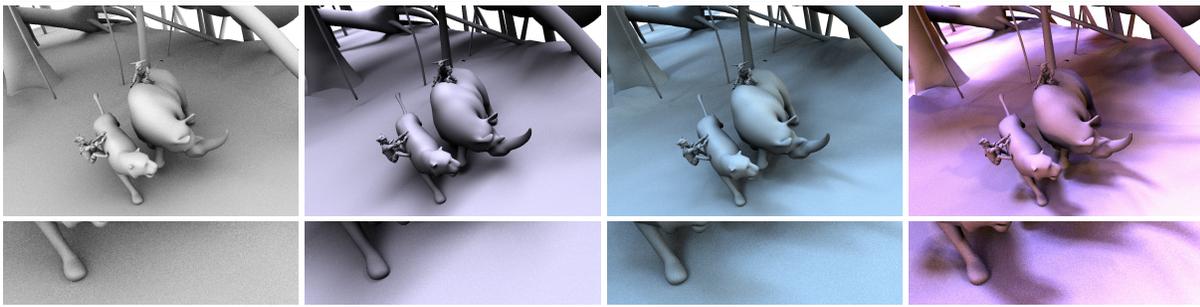
	BMW	Fly 1	Fly 2	Rhino Car
Ours	 0m:56s; RMSE = 0.0237	 2m:53s; RMSE = 0.0408	 2m:19s; RMSE = 0.0413	 4m:49s; RMSE = 0.0293
	 0m:28s; RMSE = 0.023	 0m:24s; RMSE = 0.0525	 0m:27s; RMSE = 0.0527	 0m:29s; RMSE = 0.0324
$10^2 = 100$ integration samples per pixel				
Ours	 1m:01s; RMSE = 0.0285	 3m:02s; RMSE = 0.0358	 2m:28s; RMSE = 0.0337	 5m:01s; RMSE = 0.0352
	 0m:52s; RMSE = 0.007	 0m:54s; RMSE = 0.0125	 0m:47s; RMSE = 0.0122	 0m:51s; RMSE = 0.0081
$22^2 = 484$ integration samples per pixel				
Ours	 1m:07s; RMSE = 0.0035	 3m:16s; RMSE = 0.0021	 2m:44s; RMSE = 0.002	 5m:24s; RMSE = 0.0053
	 3m:08s; RMSE = 0.0052	 3m:59s; RMSE = 0.006	 3m:04s; RMSE = 0.0064	 3m:17s; RMSE = 0.0057
$45^2 = 2025$ integration samples per pixel				

Figure 10: Convergence and performance against QMC at equal sampling rates for the scenes (but not the same lighting) in Table 1. Our discretization of $f(\omega)$ shared across pixels results in banding artifacts at low sampling rates, whereas different sampling patterns at each pixel for QMC result in noise. Our algorithm's processing time is dominated by silhouette search and it scales well with increasing sample counts, however QMC scales better with increasing geometric complexity (see Section 5).

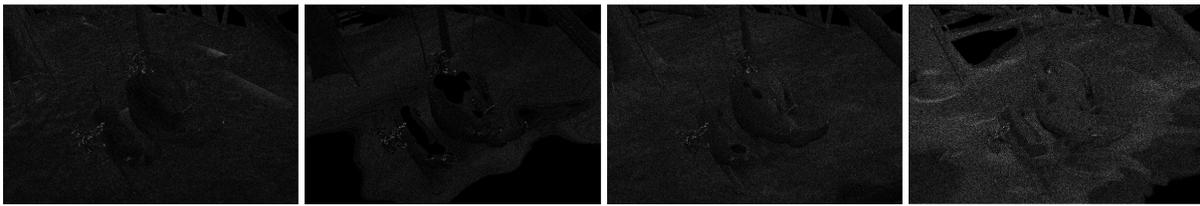
Our Result ($\sim 6\text{m}:02\text{s}$; SAT Sampling Rate: $500^2 = 250000$, regularly distributed)



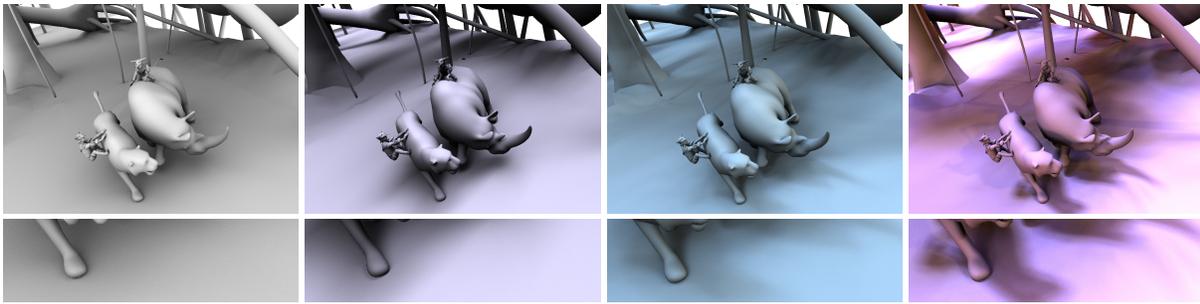
Quasi-Monte Carlo (Equal Time; $\sim 2300 - 2600$ samples)



$16\times$ absolute luminance difference image



Quasi-Monte Carlo (Equal Quality)



8500 samples / 18m:47s

6000 samples / 14m:51s

6500 samples / 15m:27s

8000 samples / 18m:41s

Figure 11: We perform an equal-time/quality comparison with importance-sampled QMC ray-tracing in the Crash scene (184.5k triangles). Our integration is independent of $f(\omega)$, and so our results compute in (approximately) equal time.

[LA05] LAINE S., AILA T.: Hierarchical penumbra casting. *Computer Graphics Forum* 24, 3 (2005), 313–322. 8

[LAA*05] LAINE S., AILA T., ASSARSSON U., LEHTINEN J., AKENINE-MÖLLER T.: Soft shadow volumes for ray tracing. *Proceedings of SIGGRAPH* (2005). 2, 3, 4, 5

[LLA06] LEHTINEN J., LAINE S., AILA T.: An improved

physically-based soft shadow volume algorithm. *Computer Graphics Forum* (2006). 2, 3

[LS10] LOOS B. J., SLOAN P.-P.: Volumetric obscurance. In *Symposium on Interactive 3D Graphics* (NY, 2010), ACM. 6

[MAAG12] MORA F., AVENEAU L., APOSTU O., GHAZANFARPOUR D.: Lazy Visibility Evaluation for Exact Soft Shad-

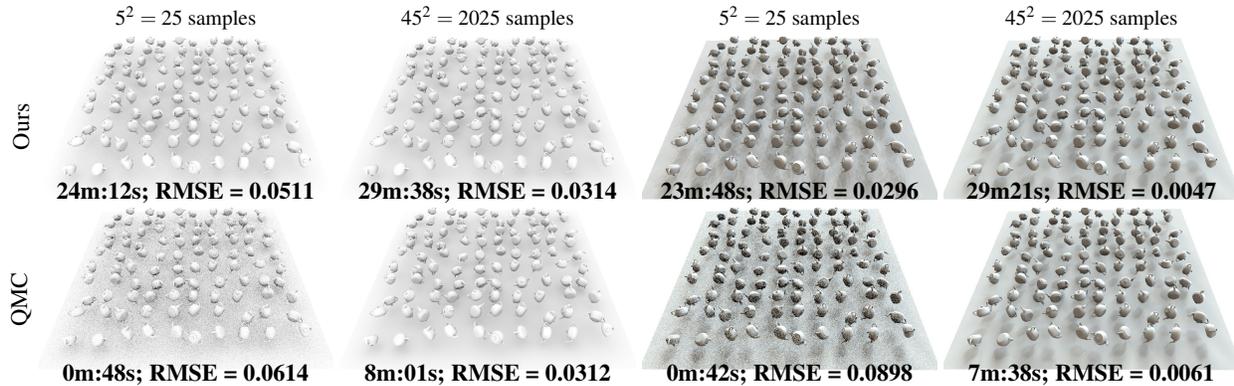


Figure 12: Stress test: comparing convergence/performance vs. QMC on a large scene (1.9 million triangles).

- ows. *Comput. Graph. Forum* 31, 1 (2012). 2
- [McG04] MCGUIRE M.: Observations on silhouette sizes. *Journal of Graphics Tools* 9, 1 (2004), 1–12. 7
- [McG10] MCGUIRE M.: Ambient occlusion volumes. In *Proceedings of High Performance Graphics* (2010). 2
- [NKLN10] NGUYEN C. H., KYUNG M.-H., LEE J.-H., NAM S.-W.: A PCA Decomposition for Real-time BRDF Editing and Relighting with Global Illumination. *Computer Graphics Forum* 29, 4 (2010), 1469–1478. 6
- [NN85] NISHITA T., NAKAMAE E.: Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), SIGGRAPH '85, ACM, pp. 23–30. 2
- [NRH03] NG R., RAMAMOORTHY R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *Transactions on Graphics* (2003). 3
- [NRH04] NG R., RAMAMOORTHY R., HANRAHAN P.: Triple product wavelet integrals for all-frequency relighting. *Transactions on Graphics* (2004). 3
- [OZ06] OLSON M., ZHANG H.: Silhouette extraction in hough space. *Computer Graphics Forum* 25, 3 (2006), 273–282. 2
- [PH03] PRAUN E., HOPPE H.: Spherical parameterization and remeshing. *ACM Transactions on Graphics* (2003). 5
- [RH01] RAMAMOORTHY R., HANRAHAN P.: An efficient representation for irradiance environment maps. In *SIGGRAPH* (NY, 2001), ACM. 2, 3, 6
- [RH02] RAMAMOORTHY R., HANRAHAN P.: Frequency space environment map rendering. In *Proceedings of SIGGRAPH* (NY, 2002), ACM. 3, 6
- [RMB07] RAMAMOORTHY R., MAHAJAN D., BELHUMEUR P. N.: A first-order analysis of lighting, shading, and shadows. *Transactions on Graphics* (2007). 8
- [Rot82] ROTH S. D.: Ray casting for modelling solids. *Computer Graphics and Image Processing* 18, 2 (Feb. 1982), 109–144. 3
- [RWS*06] REN Z., WANG R., SNYDER J., ZHOU K., LIU X., SUN B., SLOAN P.-P., BAO H., PENG Q., GUO B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. In *SIGGRAPH* (NY, USA, 2006), ACM. 2, 3
- [SGNS07] SLOAN P.-P., GOVINDARAJU N. K., NOWROUZEZHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *Pacific Graphics* (Washington, DC, 2007), IEEE. 8
- [SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *Transactions on Graphics* 21 (July 2002), 527–536. 2, 3, 7
- [Slo08] SLOAN P.-P.: Stupid Spherical Harmonics (SH) Tricks. *Game Developers Conference* (2008). 3
- [SR00] STARK M. M., REISENFELD R. F.: Exact Illumination in Polygonal Environments using Vertex Tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (London, UK, UK, 2000), Springer-Verlag, pp. 149–160. 2
- [TS06] TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *Trans. on Graphics* (2006). 3
- [VG95] VEACH E., GUIBAS L. J.: Optimally combining sampling techniques for monte carlo rendering. In *SIGGRAPH* (1995), pp. 419–428. 7
- [WP12] WOO A., POULIN P.: *Shadow Algorithms Data Miner*. Taylor & Francis, 2012. 1
- [WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. *SIGGRAPH* (1988). 8
- [WRG*09] WANG J., REN P., GONG M., SNYDER J., GUO B.: All-frequency rendering of dynamic, spatially-varying reflectance. *Transactions on Graphics* 28 (2009). 2, 3
- [ZHL*05] ZHOU K., HU Y., LIN S., GUO B., SHUM H.-Y.: Pre-computed shadow fields for dynamic scenes. In *Proceedings of SIGGRAPH* (NY, USA, 2005), ACM. 3

Appendix A: Linear SH and Vector Irradiance

Our $l = 1$ SH incident radiance is related to a vector irradiance formulation of diffuse shading. Arvo [Arv95] shows that diffuse radiance can be formulated as the dot-product of the surface normal with the first-moment of scalar irradiance, called vector irradiance:

$$\mathbf{v} = \int_{\Omega_{\mathbf{n}}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \cdot L_{in}(\mathbf{p}, \omega) d\omega, \text{ such that } L_o(\mathbf{p}) = \mathbf{n} \cdot \mathbf{v},$$

where $\Omega_{\mathbf{n}}$ is the hemisphere about \mathbf{n} . Since band- l SH basis functions are degree- l polynomials in the Cartesian coordinates of $\omega = (x, y, z)$, the $l = 1$ SH incident radiance coefficients are just a scaled permutation of the elements of \mathbf{v} , and the vector irradiance formulation is equivalent to our optimal hemispherical representation.